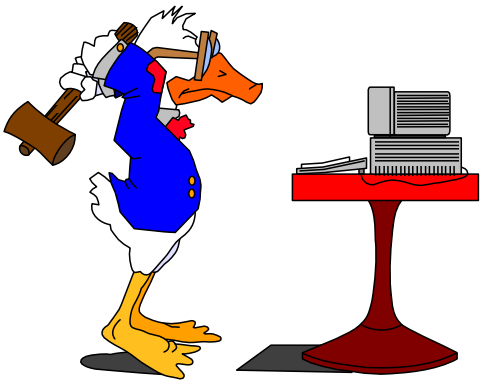
EpiInfo

Fortsättningskurs -

inmatningsformulär och dataanalys

av Mikael Åberg



Förord
Detta kursmaterial har använts i undervisningen av magisterstudenter i ämnet Folkhälsovetenskap vid Karolinska Institutet, under åren 1999-2001. Kursmaterialet är i princip intakt. En del mindre uppdateringar och revideringar har gjorts.

I kursmaterialet görs en del sidhänvisningar till Epic Info:s tryckta ”gröna” manual ”*The Epic Info Manual*”, Brixton Boorks, 1994, LONDON.

Mikael Åberg

Februari, 2010.

Allmänt om fortsättningskurs i EpiInfo

Övningsexemplen förutsätter tillgång till Epi Info filen *TI93.rec*. För varje kommando som behandlas finns en referens till Epi Info manualen.

Innehållsförteckning

Set (inställningar)

Med **set** (s.414) gör man olika inställningar i Analysis när man skall bearbeta sina data. Nedan följer de viktigaste inställningarna:

IGNORE (ta med ”missing values” eller inte ta med dem i redovisningen)

SET IGNORE = ON Ta *inte* med missing values
SET IGNORE = OFF Ta *med* missing values

Inställningar för procentberäkning
SET PERCENTS = 0FF Inga procent visas i korstabeller
SET PERCENTS = ON Rad- och kolumnprocent tas med i korstabeller

Inställningar för statistik
SET STATISTICS = OFF Inga statistikberäkningar redovisas
SET STATISTICS = ON Olika statistikberäkningar redovisas

MISSING VALUES

”internbortfall i redovisning av en fråga”

Missing values redovisas i EpiInfo med .

Om man tar med missing values eller ej regleras med set-kommandot. Det finns olika principer för om man skall redovisa tabeller och frekvenser med eller utan missing values.

I medicinska sammanhang redovisar man oftast utan att ha med missing values. I samhällsvetenskaplig tradition brukar det oftast vara kotym att man redovisar internbortfallet från exempelvis enkäter.

SELECT

Syntax : SELECT <uttryck>

SELECT (se manualen s 399 för mer information) *a*nvänds för att göra urval ur en databas (rec. datafil). Man använder sk *operatorer* för att ställa upp ett villkor som skall gälla för de egenskaper man vill analysera.

SELECT är mycket användbart och kraftfullt. Vad man måste **veta** när man använder funktionen är vilken **datatyp** man har. Om man har variabler som är **numeriska datatyper** (*eng. integers*) behöver man inte använda ””-tecknet när man sätter upp sina uttryck. Om man har variabler som är **alfanumeriska datatyper** (*eng. alpha numeric)* (tillåter både siffror och bokstäver) måste man använda ””-tecknet. Datatypen bestäms när man skapar sitt inmatningsformulär i Eped.

*Tips! Du kan alltid använda funktionen **variables** (kodschemat) i Analysis där du ser vilken datatyp en variabel har! Ett annat tips, när man arbetar med enkäter är att om möjligt definiera numeriska datatyper!*

OBS! Glöm inte att du måste **återställa** ett urval med hjälp av SELECT när du är färdig och vill göra ett helt annat urval!! En inställning ”ligger alltid kvar” tills du sagt något annat! Du återställer ett urval genom att skriva in SELECT och trycker på Enter. När du ser att det står ALL RECORDS SELECTED vid Dataset (högst upp) är urvalet återställt

Vanligaste operatorerna är:

=	exakt lika med
>	större än
<	mindre än
>=	större än eller lika med
<=	mindre än eller lika med
AND	Logiskt ‘och’. Om **alla** villkor är uppfyllda, gäller uttrycket
OR	Logiskt ‘eller’. Om ett eller flera uttryck är uppfyllda, gäller uttrycket
NOT	Logiskt ‘inte’. Om uttrycket inte är uppfyllt, gäller uttrycket

Uppgift 1
Välj ut pojkar som börjar röka när de var mellan 10-15 år.

Exempel:

select v1 = 1 and v32 >= 10 and v32 <= 15

Kommentar:

V1 är variabeln för kön. Här är pojkar kodat med 1 och flickor med 2 (numerisk). V32 är variabeln för ålder när man började röka. Här skall vi välja ett intervall där man får vara mellan 10-15 år. Här använder vi *operatorn* AND (som kan skrivas med små eller stora bokstäver. Har ingen betydelse).

Fler exempel på SELECT

```
SELECT  V1  =  1                    (enbart pojkar)
SELECT V32  <=  12                  (röka innan åldern <= 12)

SELECT V1 = 2  AND  V32 >=  13      (flickor  med rökdebue >= 13 år)

SELECT V4 <> 5                      (inga femteklassare med)
SELECT V4 = 1  OR  V4 = 3           (femte- eller niondeklassare)

SELECT                              (tar bort selektering!!!)
```

Bearbetningsexempel

```
SELECT V1 = 2                       (bara flickor)
FREQ V3                             (frekvens på flickornas födelseår)
SELECT                              (tar bort selektering)

SELECT V1 = 1                       (bara pojkar)
FREQ V3                             (frekvens på pojkarnas födelseår)
```

DEFINE

DEFINE (sid. 386) används för att skapa en ny variabel direkt i Epi Analys. Man behöver inte lägga till en ny variabel i sitt kodformulär. DEFINE används ofta tillsammans med RECODE. Först måste man tala om vilken variabel typ den nya variabeln skall ha, dvs numerisk eller alfanumerisk.

Syntax: DEFINE <variabel namn> _____ (alfanumerisk)
DEFINE <variabel namn> ### (numerisk)

LET

LET (sid. 395) används i detta stycke för att kontrollera att man inte gör fel omkodningar. Se även nästa stycke! Man måste inte skriva ut ordet "LET", det är valfritt!

Syntax: {LET} <variabel> = <uttryck>

RECODE

RECODE (sid. 404) används för att koda om eller gruppera om en variabel. Principen är att första variabeln utgör källdata från vilken omkodningen skall göras. Den andra variabeln är mottagaren av de nya koderna för en variabel.

Syntax: RECODE <Var1> {TO <Var2> {BY <Num>}} Codes

Uppgift 2
Koda om variabeln SEI så att den får namnetiketter och kör en frekvens för att se resultatet.

Exempel:

```
define nysei _____
let nysei = "FEL"
recode sei to nysei 1="SACO" 2="TCO" 3="LO" 8="LANT/FÖR" 9="" else=missing
freq nysei
```

Kommentar:

Vi skapar en ny variabel NYSEI som skall vara 10 positioner lång. Eftersom vi vill ha etiketter och skriva in text, måste vi definiera variabeln som alfanumerisk.
Vi tilldelar först variabeln NYSEI värdet "FEL".

I nästa steg ska vi koda om. Vi vet att variabeln SEI har koderna 1,2,3,4,9. Vi definierar källdatavariabeln (=den vi skall hämta våra värden *från)*, och talar om *var* vi vill lägga våra nya värden(=variabeln NYSEI). Här måste vi använda "" -tecknet för att få ut våra etiketter. Slutligen har vi sagt om variabeln SEI *inte* innehåller 1,2,3,4,9

så är det uppgift saknas, dvs *missing.* Om vi nu hade missat någon omkodning, kommer dessa ut som nysei = "FEL"!!!

Uppgift 3
Vi antar istället att vi vill *klassindela* variabeln SEI till två klasser/grupper. Gör den nya variabeln klass lite längre, exempelvis 15 tecken lång!

Exempel:

```
Define klass _______________
recode sei to klass 1,2="SACO/TCO" 3,8="LO/LANT/FÖR" else1=missing
freq klass
```

Kommentar:

Källvariabeln är fortfarande SEI och egenskaperna för SEI är fortfarande 1,2,3,8,9. Vi skapar en **ny** variabel med define kallad KLASS som är alfanumerisk . Alla som inte är 1,2,3 eller 8 blir nu missing, och tas inte med!

IF

Ibland vill man att alla data som har en speciell egenskap, skall få ett speciellt värde. Detta värde kan exempelvis vara en koefficient eller ett index för en specifik beräkning.

För att plocka ut alla som har en specifik egenskap använder vi en s.k. **IF-sats** (sid. 393).

Syntax: IF <villkor> THEN <konsekvens1> {ELSE <konsekvens2>}

LET

I det andra fallet vill vi addera ett värde 100, *givet* en egenskap, dvs 10 år. Här använder vi **LET** (sid.395) Med LET kan man använda operatorer (sid. 399)

Syntax: {LET} <variabel> = <uttryck>

Uppgift 4
Antag att vi vill att alla personer i våra databas som är exakt 10 år skall få värdet 100 adderat till sin ålder.

Exempel:

```
define calc ###
if v32=10 then let calc=v32+100
freq calc
```

Kommentar:

Först definierar vi en ny variabel CALC, i detta fall en numerisk variabel, eftersom vi vill räkna på den. Sedan väljer vi ut alla som *exakt* 10 år från åldersvariabeln v32 och adderar värdet 100. Källvariabel i detta fall är v32. Det är från den som vi hämtar våra värden och lägger de nya i variabeln CALC. Operatorn är i detta fall symbolen för addition, dvs +.

Uppgift 5

Antag att vi vill fördela svaren i Pojkar i årskurs 5, Pojkar i årskurs 7, Pojkar i årskurs 9, Flickor i årskurs 5 osv..

Här kan vi använda oss av upprepade **if**-satser.

Exempel:

```
*Skapa en ny variabel, 5 pos lång
define konars _____
let konars = "FEL"
if v1=1 and v4 = 1 then konars = "pa5"
if v1=1 and v4 = 2 then konars = "pa7"
if v1=1 and v4 = 3 then konars = "pa9"
if v1=2 and v4 = 1 then konars = "fa5"
if v1=2 and v4 = 2 then konars = "fa7"
if v1=2 and v4 = 3 then konars = "fa9"
freq konars
```

Kommentar:

Först definierar vi en ny variabel KONARS (kön, årskurs), i detta fall en alfanumerisk variabel, 5 tecken lång. Vi gör nu upprepade IF-satser för varje tänkbart alternativ. För att kontrollera att vi inte gör några fel (glömmer något alternativ) ger vi först variabeln KONARS värdet "FEL".

ROUTE

Med ROUTE (sid. 412) hanterar man hur man vill hantera *utdata*. ROUTE är användbart när man vill syra en bearbetning, exempelvis till en ny fil, som man vill ta in i ett ordbehandlingsprogram eller till en printer. ROUTE kanske oftast används när man programmerar vissa rutiner i EpiInfo.

Syntax: ROUTE <filnamn>
ROUTE SCREEN (styr till bildskärm)
ROUTE PRINTER (styr till en skrivare)

Uppgift 6
Vi vill styra ett bearbetat resultat till en textfil.

Exempel:

Route test.txt
Freq v1
route

Kommentar:

Detta exempel styr en bearbetad frekvenstabell av variabeln v1 till en textfil kallad test.txt. Notera att man måste avsluta med ROUTE ytterligare en gång för att *stänga* textfilen, annars ligger den öppen och man kan inte använda den och alla andra körningar man gör efter kommer att hamna i textfilen!!! Alternativt kan man skriva CLOSE som har samma funktion och stänger filen test.txt

WRITE

WRITE (sid426) tillsammans med ROUTE använder man om man vill t.ex. plocka ut data som har en speciell egenskap och lägga det i en ny datafil.

Syntax: WRITE {RECFILE} {<variabel namn>}

Uppgift 7
Välj ut pojkar som börjar röka när de var mellan 10-15 år och lägga dem i en ny datafil.

Exempel:

select v1 = 1 and v32 >= 10 and v32 <= 15
erase pojk1015.rec
route pojk1015.rec
write recfile

Kommentar:

Vi gör samma urval med **select** som i första övningen. Vi har lagt in kommandot **erase** (radera) som skall radera eventuellt tidigare versioner av datafilen *pojk1015.rec* som vi tillverkat. Detta är viktigt att ha i åtanke: lägger man inte in **erase** kommer nya data att läggas till datafilen *pojk1015.rec.* Kör man proceduren fler gånger riskerar att få dubbletter i datafilen (varje post blir dubblerad!) Notera att vi inte behöver avsluta med ROUTE här, eftersom den nya datafilen stängs automatiskt efter det att den nya datafilen skapats! Att skapa nya rec-filer är en mycket effektiv metod, om kör på stora datafiler och vill dela upp sitt material, eller dela upp det efter speciella egenskaper.

REPORT

REPORT (sid 407) är en funktion för att skapa fasta, återkommande rapporter. Funktionen är aningen komplicerad att använda och används i första hand för att rapportera in data, där inläsningsprocedurerna är de samma men att data ändras över tiden. Ett användbart område är rapporter från en kontinuerlig olycksfallsrapportering

Man skapar en rapportrutin i två steg, (1) skapa en .rpt-fil med de variabler som skall ingå (2) kör denna rapportrutin i EpiInfos Analysis. Programslingan kan skapas i en vanlig enkel ASCII ordbehandlare
(exempelvis den som finns inbyggd i EpiInfo, Analysis(EDIT) eller i EPED. Tänk bara på att du måste spara den med **rpt** som filslutsändelse!.Rapporten körs sedan i EpiInfo ANALYSIS med start kommandot REPORT.

Syntax: REPORT <rapportfilens namn>

Uppgift 8
Gör en enkel rapport som letar upp och räknar hur många som finns i varje socio-ekonomisk grupp.

Exempel:

#uses domsei
#foreach domsei
[^domsei] kommer från socialgrupp se (SEI-klassifikation)
#endfor

Kommentar:

Ovanstående exempel skapas i ordbehandlaren EDIT i Analysis. Man skapar den genom att skriva i **edit rap93.rpt**. Viktigt är att man måste definiera vilka variabler man vill ha i rapporten. Detta görs med kommandot #uses, följt av variabeln domsei från datafilen. I nästa rad letar programslingan upp variabeln domsei och dess koder och summerar dessa samt att en klartextbeskrivning ges. Sista raden anger att här skall den sluta leta och börja om igen och fortsätta genom hela datafilen (en sk loop). Gå ur EDIT och spara. Kör programmet genom att skriva **report rap93.rpt**

Avancerad användning av REPORT

Ibland kan det vara användbart att bygga s.k. fasta rapporter. En fast rapport är användbar om man har data som skall rapporteras kontinuerligt. Ett exempel är olycksfallsregistrering, där man kanske varje månad vill köra fram statistikrapporter från databasen. Nedanstående exempel visar hur slutresultatet från en sådan rapportfil kan se ut:

```
*** RAPPORT ***
RESULTAT – Från Olycksfallsprojektet
Denna analys avser:  samtliga
Filnamn: aldrebo9.rec
Program: Epi Info ver 6.04b
Tot antal formulär: 194
Rapportdatum: 08/31/98
av Mikael Åberg
============================================================

Kön:

                                          Antal    Procent
man                                          55         28
kvinna                                      139         72
Uppgift saknas                                0          0
SUMMA                                       194        100

Tidpunkt när olyckan inträffade:

                                          Antal    Procent
morgon                                       37         19
förmiddag                                    22         11
eftermiddag                                  54         28
kväll                                        45         23
natt                                         34         18
uppgift saknas                                2          1
SUMMA                                       194        100

Var inträffade fallet:

                                          Antal    Procent
kök                                           7          4
badrum                                       17          9
sovrum                                       94         48
vardagsrum                                   20         10
hall                                         27         14
ute                                           1          1
annat                                        17          9
uppgift saknas                               11          6
SUMMA                                       194        100
```

Tekniken bakom

Ovanstående exempel ger en enkel, användarvänlig rapport som skapas som en textfil. Denna textfil kan sedan enkelt importeras till exempelvis MS Word.

Procedurfilen är en s.k. pgm-fil som i detta fall heter OLYXMAIN.PGM. Från denna pgm-fil görs anrop till rpt-filer.

```
*Detta är filen OLYXMAIN.PGM som styr hela bearbetningen till rapporter.
*av Mikael Åberg, Utvecklingsenheten i Tibro, tfn 0504-18553.
*mikael.aberg@primnet.se

*Läs in filen EpiInfo datafilen, ÄLDREBO9.REC
read aldrebo9

*Inställningar med SET-kommandot
set ignore = off

:START
*Styr resultat till textfiler
PICKLIST
  "Samtliga" goto SAMT
  "Män" goto MAN
  "Kvinnor" goto KVINNOR
  "KÖK" goto KOK
  "BADRUM" goto BADRUM
  "SOVRUM" goto SOVRUM
  "VARDAGSRUM" goto VARUM
  "HALL" goto HALL
  "UTE" goto UTE
  "ANNAT" goto ANNAT
  "Avsluta" goto AVSLUT
END

:SAMT
erase aldsam9.txt
route aldsam9.txt
set page = 60,150
let kon = "Samtliga"
***************************** ANALYSERA **********************************
:ANALYS
*Räkna antal enkäter!!!
define antal ### cumulative
antal = antal + 1
process
type "               *** RAPPORT ***                        "
type "RESULTAT - PROJEKT BEN SOM BÄR          "
type "Denna analys avser:  samtliga                          "
type "Filnamn: aldrebo9.rec "
type "Program: Epi Info ver 6.04b"
type "Tot antal formulär: @antal "
type "Rapportdatum: @systemdate"
type "av Britta Larsmark"
type "============================================================="
type " "

type "Kön: "
report sex

*type "Födelseår årtal 19xxr:"
*report klass

*Fråga 2
type "Tidpunkt när olyckan inträffade: "
```

```
report v2

*Fråga 3
type "Var inträffade fallet:"
report v3

*Fråga 4
type "Föll:"
report v4

*Styr resultatet till bildskärm igen
route
select
goto START
:AVSLUT
goto SLUT
```

I koden ovan anropas rapportfiler från rapportgeneratorn. Dessa rapportfiler heter *report klass, report v2 osv.* Nedan visas ett exempel på hur en sådan rapportfil ser ut (sex.rpt):

```
#uses sex

                                          \9  Antal\9Procent
man                                      \9[1]:7\9{[1]*100/[]}:7.0
kvinna                                   \9[2]:7\9{[2]*100/[]}:7.0
Uppgift saknas                           \9[.]:7\9{[.]*100/[]}:7.0
SUMMA                                    \9[]:7\9{[]*100/[]}:7.0
```

KOMMENTAR till rapportfilen sex.rpt:

# uses = talar om att vi skall använda variabeln sex från datafilen
man = i databasen är man kodat med 1
kvinna =kodat med 2
Uppgift saknas=kodat med .
\9 och :7 är koder för formatering och layout

Uppgift 9

(Innan du gör denna uppgift, läs mer under avsnittet *Om att skriva programfiler (PGM-filer)* och avsnittet längre fram.) Bygg upp en fast rapport med 4 variabler från databasfilen *ti93.rec* . Använd ovanstående som underlag och referens när du bygger upp proceduren. Tänk på att följande filer måste finnas i samma katalog för att det skall fungera:

- 4 st. rpt-filer
- en pgm-fil

SAVE och RUN

SAVE och RUN (sid. 412) används för att spara och respektive köra programfiler.

Om man har skrivit några kommandon och kört dessa kan man spara de sista 20 man kört med hjälp av kommandot SAVE.

Dessa program kan man sedan redigera och köra genom kommandot RUN. Progamfilerna slutar på *.PGM.

Syntax: SAVE <filens namn>
Syntax: RUN <programfilens namn>

Uppgift 10
Spara de senaste körningarna med hjälp av kommandot SAVE, gör ändringar med hjälp av editorn EDIT och kör programfilen med RUN.

Exempel:

Save testprog.pgm
Edit testprog.pgm
(gör nu dina ändringar och spara med F10)
Run testprog.pgm

Kommentar:

Spara kommandona till en programfil som heter TESTPROG.PGM. Gå sedan in i editorn i Analysis och gör ändringar i kommandona. Spara med F10. Kör sedan programfilen med kommandot RUN.

Om att skriva programfiler (PGM-filer) EpiInfo
EpiInfo använder ett Basic liknande programmeringsspråk. Alla Epi Info programfiler slutar på .pgm. Man läser in dem i ANALYSIS med kommandot *run*. I programfilerna kan man inkludera DOS-kommandon. Avancerade programfiler kan också byggas och köras i s.k. batch-miljö i DOS.

Om man vill lära sig och skriva programprocedurer i Epi Info är det lämpligt att man använder en EDITOR (=enkel ordbehandlare). Notera att det måste vara en ordbehandlare av ASCII-typ. Använder man exempelvis MS Word kommer Epi Info inte att kunna läsa filerna eftersom en rad olika styrtecken kommer att sparas nedsom EpiInfo inte förstår. I Analysis finns en inbyggd EDITOR (kallad EDIT) som man kan använda. Man kommer in i den igenom att skriva EDIT vid kommandoraden. Ett tips: om man skall skriva en epi info procedurfil är det lämpligt att man skriver hela programfilens namn på engång i samband med att man öppnar den. I Analysis på kommandoraden skriver man *edit_procedur.pgm. (*Tecknet _ markerar ett mellanslag.) Spara filen genom att trycka F10 för att gå ur.

Programeringstips
Ett sätt att spara tid när man bygger upp sina Epi Info procedurer är att först köra dem på skärmen som vanliga kommandon. När man har tagit fram de procedurer som man tycker fungerar och vill använda, använder man kommandot SAVE och sparar ner de kommandon man skrivit till en fil. Denna fil kan med fördel döpas som en .pgm-fil! Därefter redigerar man denna fil i ordbehandlaren EDIT som finns i Analysis.

Resurser om Epi Info

På Internet finns några bra resurser där man kan få hjälp när man arbetar med EpiInfo. Bra resurser är Epi Infos hemsida på adressen:

http://www.cdc.gov/epiinfo/Epi6/ei6.htm

Länk där man hämtar hem programmet:
ftp://ftp.cdc.gov/pub/Software/epiinfo_dos/

Vid installation, ladda först ner instruktionsfilen,
ftp://ftp.cdc.gov/pub/Software/epiinfo_dos/getthese.txt